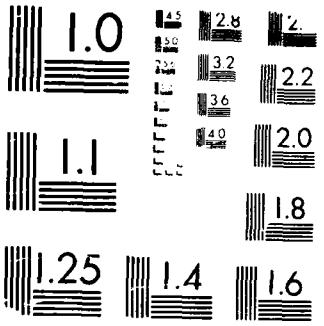
AD-A194 567 DESIGN OF A SELF-TIMED VLSI MULTICOMPUTER COMMUNICATION CONTROLLER(U) MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE L.. W J DALLY ET AL. NOV 87 1/1

UNCLASSIFIED VLSI-MEMO-87-423 N00014-80-C-0622 F/G 12/7 NL



MICROCOPY RESOLUTION TEST CHART

VLSI Memo No. 87-423
November 1987

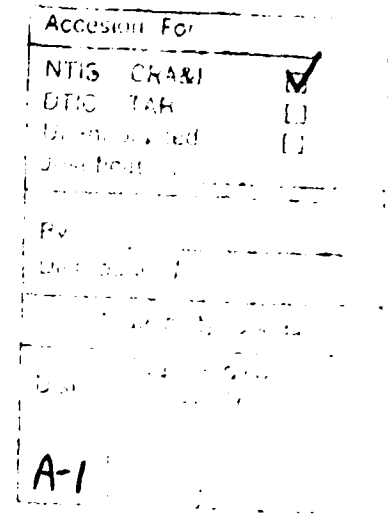
DESIGN OF A SELF-TIMED VLSI MULTICOMPUTER COMMUNICATION CONTROLLER

William J. Dally and Paul Song

Abstract

We describe the design of the network design frame (NDF), a self-timed routing chip for a message-passing concurrent computer. The NDF uses a partitioned data path, low-voltage output drivers, and a distributed token-passing arbiter to provide a bandwidth of 450Mbits/sec into the network. Wormhole routing and bidirectional virtual channels are used to provide low latency communications, less than 2μs latency to deliver a 216bit message across the diameter of a 1K node machine. To support concurrent software systems, the NDF provides two logical networks, one for user messages and one for system messages, that share the same set of physical wires. To facilitate the development of network nodes, the NDF is a design frame. The NDF circuitry is integrated into the pad frame of a chip leaving the center of the chip uncommitted.

Microsystems Research Center Room 39-321
Massachusetts Institute of Technology
Cambridge Massachusetts 02139
Telephone (617) 253-8138



Acknowledgements

This work was supported in part by the Defense Advanced Research Projects Agency under contract numbers N00014-80-C-0622 and N00014-85-K-0124, and by a National Science Foundation Presidential Young Award, with matching funds from the General Electric Corporation.

Author Information

Dally, Song: Artificial Intelligence Laboratory and Laboratory for Computer Science, MIT, Cambridge, MA 02139. Dally: Room NE43-419, (617)253-6043; Song: Room NE43-415, (617)253-6048.

Design of a Self-Timed VLSI Multicomputer Communication Controller[1]

William J. Dally and Paul Song

Artificial Intelligence Laboratory and
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

We describe the design of the network design frame, (NDF), a self-timed routing chip for a message-passing concurrent computer. The NDF uses a partitioned data path, low-voltage output drivers, and a distributed token-passing arbiter to provide a bandwidth of 450Mbits/sec into the network. Wormhole routing and bidirectional virtual channels are used to provide low latency communications, less than $2\mu s$ latency to deliver a 216bit message across the diameter of a 1K node machine. To support concurrent software systems, the NDF provides two logical networks, one for user messages and one for system messages, that share the same set of physical wires. To facilitate the development of network nodes, the NDF is a design frame. The NDF circuitry is integrated into the pad frame of a chip leaving the center of the chip uncommitted.

1 Introduction

The critical component of a concurrent computer is its communication network. Many algorithms are communication rather than processing limited. Fine-grain concurrent programs execute as few as 10 instructions in response to a message [4]. To efficiently execute such programs the communication network must have a latency no greater than about 10 instruction times, and a throughput sufficient to permit a large fraction of the nodes to transmit simultaneously. Low-latency communication is also critical to support code sharing and garbage collection across nodes.

This paper describes the design of a self-timed communication controller, the network design frame (NDF). The NDF performs routing and flow-control to perform end-to-end delivery of messages between any two nodes in a k-ary n-cube network [20]. The NDF provides a 450Mbits/sec bandwidth into the network with a maximum latency of $2\mu s$ to send a 6 word (216 bit) message between the most distant nodes of a 1024 node mesh-connected machine. To achieve this level of performance, the NDF design uses low-voltage (1V) output drivers [15]. A distributed token-passing arbiter is used to reduce arbitration time, and a partitioned self-timed control circuit minimizes the control path delays. Virtual channels are used to implement two completely separate logical networks sharing a single set of physical wires.

We plan to integrate the NDF into the pad frame of a chip (hence the name) reserving the center of the chip for the logic used to implement the network node. To make this integration possible we have reduced the size of the router by implementing separate dimension datapaths and eliminating the large crossbar switch used on previous designs such as the torus routing chip TRC [6].

The NDF builds on previous work done in the development of the TRC. Both chips use low-dimensional k-ary n-cube topology to achieve low latency communication [7], and use the technique of virtual channels [5]. Cut-through or wormhole routing is used by both chips to make latency the sum of message length and distance rather than the product of these two components [14] [21]. The self-timed design techniques presented in this paper build upon the work of Seitz [19], Martin [16], Molnar [17], Chu [3], and Hollaar [12].

In Section 2 we present the system architecture of a network constructed using NDFs. The organization of an individual NDF is described in Section 3. Section 4 describes the logic design of critical components. Performance figures for the NDF are presented in Section 5.

2 Architecture

The NDF is a *design frame* [2] that facilitates the integration of new node types into a message-passing system. The logic of the NDF is integrated into the pad-frame of a chip (Figure 1) leaving the center of the chip for node-specific logic. The node interfaces with the network over two unidirectional 11-bit connections with unidirectional control signals. The two priority levels are multiplexed onto each connection. Any node type that supports the network's simple protocol can be integrated into the NDF.

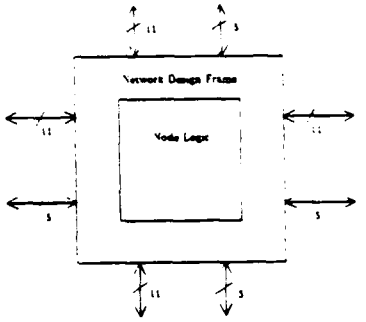


Figure 1: The NDF is integrated into the pad-frame of a chip. The center of the chip is left uncommitted to be used for the logic of different types of processing nodes.

Figure 2 shows a heterogeneous message-passing system constructed using NDFs. The NDF provides communication services for symbolic processors [8], arithmetic processors [10], I/O devices, and memories. For a node, A, to send a message to another node, B, A feeds the message into its local NDF in 9-bit *flits* at 50MHz . Appended to each flit are a parity bit and a tail bit for a total of 11bits. The first two flits are interpreted as the absolute address (X,Y) of the destination. The remaining flits are the message text. The last flit of the message text has its tail bit set. Node A's NDF converts the absolute address to an relative address and selects the first channel of the route. Each flit of the message is forwarded over this channel as soon as it is received. Subsequent NDF's along the path from A to B update the relative address, select an output channel, and forward received flits over the output channel until a

[1]The research described in this paper was supported in part by the Defense Advanced Research Projects Agency under contracts N00014-80-C-0622 and N00014-85-K0124, in part by an NSF Presidential Young Investigator Award, and in part by a grant from General Electric.

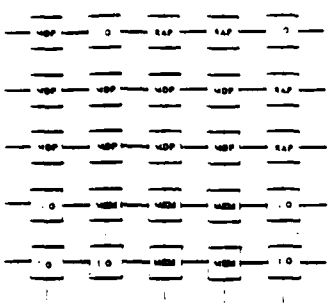


Figure 2: A heterogeneous message-passing system constructed from NDFs. The NDFs provide end-to-end communication services for the different node types.

tail flit is detected. The NDFs operate at 50MHz with an input to output delay of 20ns . Routing, arbitration, and buffering are performed entirely within the NDFs. There is no interference with intermediate nodes.

The NDF implements a bidirectional mesh topology. Our first NDFs support 2-D meshes. The design is easily extended to higher dimensions. Using a mesh (a k-ary n-cube with the ends open - Figure 3A) rather than a torus (a k-ary n-cube with the ends wrapped around - Figure 3B) reduces the bisection width (allowing wider channels) and simplifies deadlock avoidance but increases diameter and breaks symmetry. The mesh, because its channels are partially ordered, performs deadlock free routing without the use of virtual channels. A torus would require two virtual channels for each physical channel, doubling the size of the controller and the number of control lines required. The price of this simplicity is increased diameter (kn in the mesh vs. $kn/2$ in the torus) and a loss of symmetry that results in non-uniform loading of channels. For large k, the loading of a center channel is $k/4$ times the loading of an edge channel.

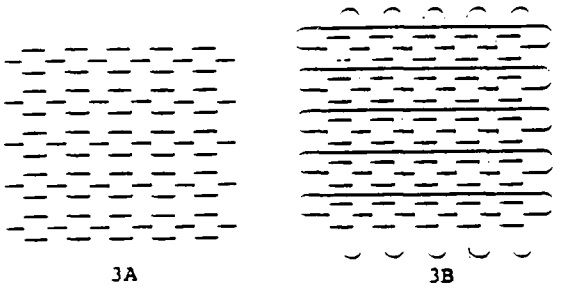


Figure 3: The NDF implements a bidirectional mesh (A) rather than a torus (B). This allows it to provide deadlock-free routing without the use of virtual channels.

Figure 4A shows the pinout of a connection between two NDFs. Four virtual channels (two directions × two priority levels) are multiplexed on eleven bidirectional lines. A token-passing protocol using the single arbitration line, ARB, controls access to the data lines. At a given instant one of the two NDFs, A, has the token (and hence control of the *data lines*). B may request the token by driving ARB high. The token is passed when A drives the line low (Figure 4B). A and B then reverse roles. On reset, the south or west side of a connection is given the token. As shown in Figure 4C, once a NDF has control of the connection, a flit is transferred by driving an R/A line high. The channel is released after a short hold time, viz. there is no acknowledge. When the receiving NDF is ready to accept another flit over the channel, it pulls the corresponding

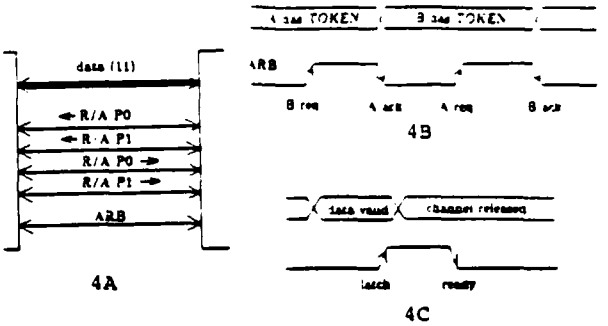


Figure 4: Pinout of a connection between the East (E) port of chip A to the West (W) port of chip B. The connection multiplexes four virtual channels over a single set of 11 bidirectional wires using 4 request/acknowledge lines (one per channel) and a single arbitration line.

R/A line low.

Collisions are resolved by blocking messages in the network. No messages are dropped. When a message requires a channel that is already in use, the head flit of the message is stopped. Following flits continue to advance until two flits are buffered on each node. When the channel becomes available, the message is advanced starting with the head, and the message again spreads out so that each flit is on a *different* node. A blocked message retains control of all channels between its head and tail, and may in turn block other messages. However, because the dependency graph of the routing function is acyclic, message blocking is guaranteed not to cause deadlock [5]. This protocol results in a throughput that approaches half the network capacity [7].

The NDF uses virtual channels [5] to provide two logically separate networks (priority levels) on the same physical wires. Latency sensitive messages (e.g., forwarding or combining) can be run through one network avoiding the majority of traffic traveling in the other network. Also, system messages needed to relieve congestion can be transmitted through a network congested with user messages. For example, when the message queue of a node overflows causing packets to back up into the network, system messages are required to transfer part of the message queue to another node or to a secondary storage device. When two classes of communication service are required (e.g., fast/slow, system/user, combining/noncombining), implementing two logical networks on one set of physical channels is an attractive alternative to building two separate networks as is being done in the RP3 [18].

On power-up, the NDF performs a self-test and calculates the absolute address of each node in the network. During routing, these node addresses are subtracted from absolute message addresses to compute relative message addresses.

3 Router Organization

The NDF contains separate routing logic for two priority levels. Each priority level consists of two dimensions as shown in Figure 5. Each dimension of the NDF receives inputs from either of two directions in the dimension or from a *higher* dimension. The messages are either forwarded along the same direction in the given dimension or down to a *lower* dimension.

Partitioning the datapath into dimensions and priority levels allows all routing to be performed by two-way switches. Previous routing chips [6] have used full crossbar switches. A 9 × 9 by 11-bit wide crossbar switch would be required to implement the NDF. This switch would require ≈ 20 times the area of the 12 11-bit wide two-way switches used in the NDF. The partitioned data path also results in a router that is ≈ 3

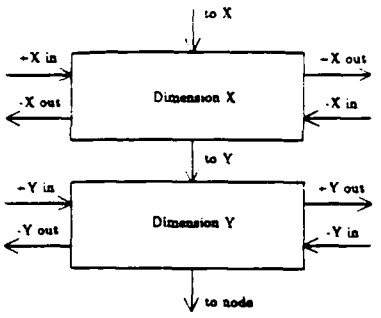


Figure 5: NDF block diagram. An NDF consists of two priority levels each of which is composed of two dimensions.

times faster because both the depth and the fanout of the data paths are minimized.

Figure 6 shows a single dimension of the NDF. The basic building blocks are the Routing Control Unit (RCU), Output Control Unit (OCU), and the Bus Arbitration Unit (BAU).

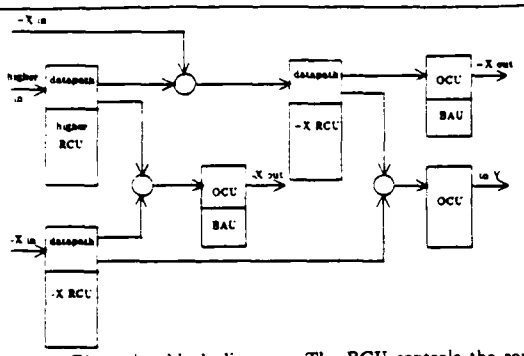


Figure 6: *Dimension block diagram.* The RCU controls the routing of messages. The OCU and BAU control the transfer of flits between chips.

Each RCU makes a two-way routing decision based on a zero or sign check of the first flit of a message. The RCU sets a switch and then competes with other RCUs for access to the selected output channel. When this arbitration is won, the RCU sends data to the selected OCU. All of the flits of the message up to and including the tail flit are routed across the switch. The RCU holds the channel for the entire duration of the message.

The OCU controls the internode communication and arbitrates between the two priority levels for access to the physical channel. There is an OCU for each of the four directions (+X, -X, +Y, -Y). The OCU interfaces to an RCU on the neighboring node and performs the request side of the protocol for transferring flits between chips. When the OCU receives a flit, it makes a request to the BAU for access to the channel. When the BAU acknowledges the request, the OCU competes with the other priority level for the use of the lines. When this arbitration is won, the OCU drives the data onto the wires and raises the appropriate R/A line to transfer the data. Arbitration is performed for each flit of a message allowing the physical channel to be multiplexed on a flit by flit basis between four virtual channels (2 priority levels × 2 dimensions).

The BAU implements the token passing protocol that arbitrates between the nodes for access to the channel. The BAU interfaces to the local OCUs and (via the ARB line) to the BAU on the neighboring node. The BAU receives requests from the OCUs for access to the channel. If the BAU holds the *token*, an acknowledgment is given immediately. Otherwise the BAU requests the token from the neighboring BAU and an acknowledgment is given when the token is acquired.

Recall that the routing data is contained in the first two flits of the message. The first flit of the message provides the relative X address (*in two's complement*) of the destination node (*number of hops in the* X dimension) and the second flit provides the relative Y address of the destination node.

A message arriving from the +X input is examined by the +X RCU. If the destination address is zero, the address flit is stripped and the RCU forwards the rest of the message to the next dimension (Y). Otherwise, the address flit is decremented and the rest of the message continues in the +X direction. Likewise, a message arriving into the -X direction will either continue in the -X direction or be forwarded into the Y dimension.

A message arriving from a *higher* dimension into dimension, D, is routed to one of three possible output directions. A sign check is initially performed on the header flit. If it is negative, the message is forwarded to the -D OCU. If it is positive, the message is sent to the +D RCU which performs a zero check and forwards the message accordingly.

When a message is forwarded to a lower dimension, the header flit is stripped. The new header flit contains the relative address of the destination node in the lower dimension. This allows all dimensions to be identical. The input register of the node logic accepts messages from the lowest dimension and the output register of the node logic transmits messages into the highest dimension.

4 Logic Design

Decrementer

The 9-bit incrementer/decrementer is the slowest element in the NDF's data path. An early design of the NDF used a Galois counter (the combinational equivalent of a linear-feedback shift register) [1] for the decrementer. Relative addresses were represented as polynomials over GF(2) and the decrement was performed in a single exclusive-or delay. However, converting absolute addresses to relative polynomial addresses proved prohibitively expensive and this approach was abandoned.

Figure 7 shows one bit of the NDF decrementer. A precharged Manchester carry chain with carry lookahead across the low-order five bits is used to achieve a worst-case delay of 6ns . A carry completion signal is generated by detecting the arrival of the carry at a bit where it will not be propagated further. This completion signal triggers the OCU when the output of the decrementer is valid. A multiplexer selects between the original and decremented value. This multiplexer allows the decrementer to be run in parallel with the RCU.

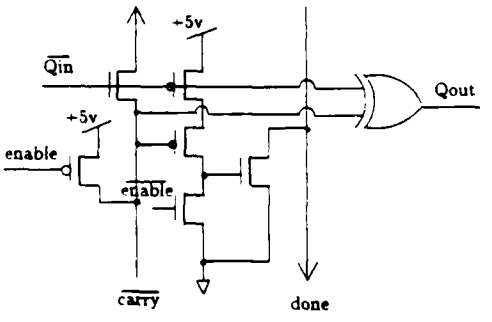


Figure 7: NDF Decrementer. One bit of the decrementer is shown. A Manchester carry chain with carry lookahead is used to achieve a delay of 6ns .

The token-passing protocol for allocating control of a channel is performed by BAUs on neighboring nodes communicating over the ARB line. Figure 8 shows the state diagram for the BAU. BAU A (South or West) starts in state 00 (with the token), and BAU B (North or East) starts in state 01 (without the token). If BAU A receives a request from

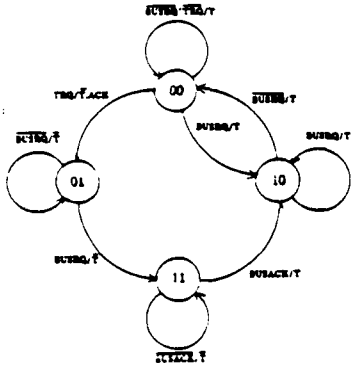


Figure 8: Bus Arbitration Unit (BAU) state diagram.

its local OCU (BUSRQ) for use of the channel, it acknowledges immediately (since it has the token) and enters state 10 (channel in use). After the OCU transmits its flit, it lowers the request and BAU A returns to state 00. If BAU B receives a BUSRQ, it requests the token (TRQ) from A over the ARB line (state 11). When A is in state 00 (channel not in use), the token will be transferred causing A to enter state 01 (without token), and B to enter state 11 (with token, channel in use). Because the BAUs are self-timed, an arbiter must be used in state 00 to select between the request lines TRQ, and BUSRQ. The logic diagram for the BAU is shown in Figure 9.

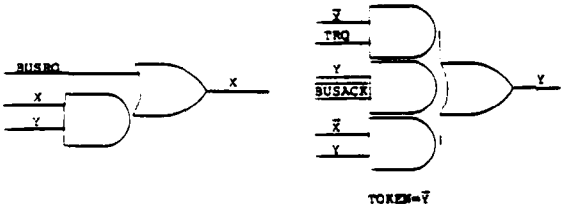


Figure 9: Bus Arbitration Unit (BAU) logic diagram.

The token-passing protocol reduces message latency, by eliminating the need to arbitrate between nodes in the frequent case that the local BAU has the token. The channel may multiplex flits from the two priority levels of a single node without need of internode arbitration. Multiplexing flits from opposite directions, however, incurs the delay of transferring the token between nodes. This mechanism retains the flexibility of multiplexing flits from four sources over the channel without paying a performance penalty on every flit.

5 Performance

Critical paths in the NDF have been simulated with SPICE. Using typical model parameters for a 2μ process, the input to output delay for a flit traveling in the same direction when the token is present on the node is 20ns . The round-trip delay between the OCU and the input latch of the next node is 20ns giving an operating frequency of 50MHz and a bandwidth per channel of 450Mbits/sec . Delays of the individual NDF subsystems are tabulated below.

decrementer	6ns
zero check	2ns
latch	1.5ns
pads	4ns
OCU	3ns
RCU	4ns

6 Conclusions

We have designed the Network Design Frame (NDF), a *self-timed* VLSI multicomputer communications controller that provides end-to-end (datagram) communication in multicomputer networks. The bandwidth of an NDF channel is 450Mbits/sec and the delay through an NDF is 20ns giving a maximum latency of $2\mu s$ for sending a 6-word (216-bit) message across the diameter of a 32-ary 2-cube (1024 nodes). All routing, flow-control, and arbitration are performed by the NDFs along the route. No memory bandwidth or CPU time on intermediate nodes is used for routing. The NDF is integrated into the pad frame of a chip to facilitate interfacing new node types with a standard network.

The NDF incorporates a number of design innovations that improve performance and reduce area compared to previous router designs [6]

- A partitioned data path (Figure 5).
- Bidirectional communication channels.
- Low voltage swing output pads [15].
- Bidirectional request/acknowledge lines.
- A distributed token-passing arbiter [16].
- Two logical networks multiplexed on one physical network.

The logical design and layout of the NDF are complete and the chip has been submitted for fabrication. We expect to test prototype NDFs in October 1987.

An area for further investigation is the development of an efficient adaptive routing algorithm for this class of networks. The NDF and the TRC both use deterministic or oblivious routing – viz. the route does *not depend on network traffic*. An adaptive routing algorithm would give superior performance under conditions of heavy, non-uniform traffic. Valiant has described an algorithm that randomizes traffic [22]. This approach, however, destroys any locality present in the communication pattern. The Connection Machine [11] and HEP [13] use *desperation routing* to route messages around congestion. However this approach has not been proved to be deadlock and livelock free.

We intend to use the NDF in the construction of a message-passing concurrent computer [9]. Its use, however, is not limited to multicomputers. Networks based on the NDF are an efficient communication mechanism for connecting subsytems in most digital systems. As today's board level subsystems are integrated into single chips, direct communication networks based on controllers such as the NDF offer higher bandwidth, higher connectivity, greater concurrency, and lower power dissipation than buses at the expense of a slightly greater latency. We expect to see such networks used to connect the processors, memory modules, and I/O devices of serial computers, and to connect the subsystems of special purpose digital processors.

References

[1] Blahut, Richard E., *Theory and Practice of Error Control Codes*. Addison-Wesley, 1983, pp. 65-90.

[2] Borriello, G., Katz, R.H., "Design Frames: A New System Integration Methodology," *1985 Chapel Hill Conference on VLSI*, Henry Fuchs, Ed., Computer Science Press, 1985, pp. 19-34.

[3] Chu, Tam-Anh and Glasser, Lance A., "Synthesis of Self-timed Control Circuits from Graphs: An Example", *Proceedings of International Conference on Computer Design*, 1986, pp. 565–571.

[4] Dally, William J., *A VLSI Architecture for Concurrent Data Structures*, Ph.D. Thesis, Department of Computer Science, California Institute of Technology, Technical Report 5209:TR:86, 1986.

[5] Dally, William J. and Seitz, Charles L., " Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, Vol. C-36, No. 5, May 1987, pp. 547-553.

[6] Dally, William J. and Seitz, Charles L., "The Torus Routing Chip," *J. Distributed Systems*, Vol. 1, No. 3, 1986, pp. 187-196.

[7] Dally, William J. "Wire Efficient VLSI Multiprocessor Communication Networks," *Proceedings Stanford Conference on Advanced Research in VLSI*, March 1987, pp. 391-415.

[8] Dally, William J. et.al., "Architecture of a Message-Driven Processor," *Proceedings of the 14th Symposium on Computer Architecture*, June 1987, pp. 189-196..

[9] Dally, William J., "The J-Machine: A Concurrent VLSI Message-Passing Computer for Symbolic and Numeric Processing," to appear.

[10] Fiske, Stuart, and Dally, William J., "The Reconfigurable Arithmetic Processor," to appear.

[11] Hillis, W.D., *The Connection Machine*, MIT Press, 1985.

[12] Hollaar, Lee A., "Direct Implementation of Asynchronous Control Units", *IEEE Transactions on Computers*, Vol C-31., No 12, 1982, pp. 1133-1141.

[13] Jordan, Harry F., "Performance Measurements on HEP - A Pipelined MIMD Computer," *Proceedings of the 14th Symposium on Computer Architecture*, 1983, pp. 207-212.

[14] Kermani, Parviz and Kleinrock, Leonard, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, Vol 3., 1979, pp. 267-286.

[15] Knight, Tom, and Krymm, Alex, "Self Terminating Low-Voltage Swing CMOS Output Driver," *Proc. CICC-87*, 1987.

[16] Martin, Alain J., "The Design of a Self-Timed Circuit for Distributed Mutual Exclusion", *Proceedings of Chapel Hill VLSI Conference*, 1985,pp. 245–260.

[17] Molnar, "Synthesis of Delay Insensitive Modules," *Proceedings of Chapel Hill VLSI Conference*, 1985, pp. 67-86.

[18] Pfister, G.F. et. al., "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture", *Proceedings ICPP*, 1985, pp. 764–771.

[19] Seitz, Charles L., "System Timing" in *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway, Addison-Wesley, 1980, Ch. 7.

[20] Seitz, Charles L., "Concurrent VLSI Architectures," *IEEE Transactions on Computers*, Vol. C-33, No. 12, December 1984, pp. 1247-1265.

[21] Seitz, Charles L., et al., *The Hypercube Communications Chip*, Dept. of Computer Science, California Institute of Technology, Display File 5182:DF:85, March 1985.

[22] Valiant, L.G., "A Scheme for Fast Parallel Communication," *SIAM J. Computing*, Vol. 11, No. 2, May 1982, pp. 350-361.

END

DATE

FILMED

8-88

DTIC